

Build 1: Control A DC Motor

Description:

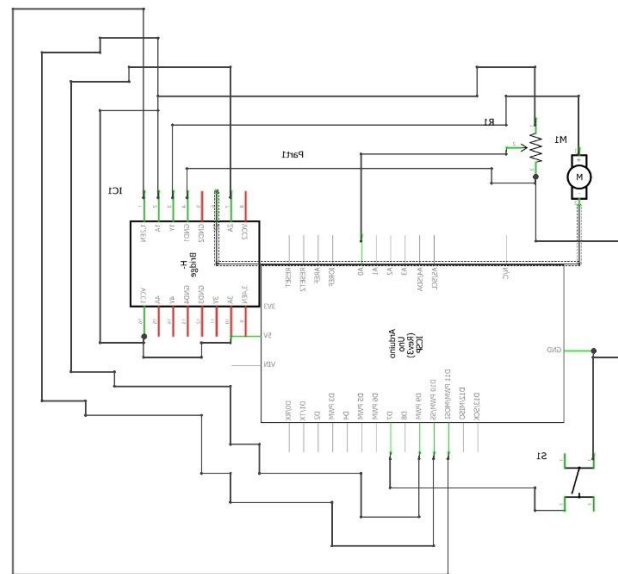
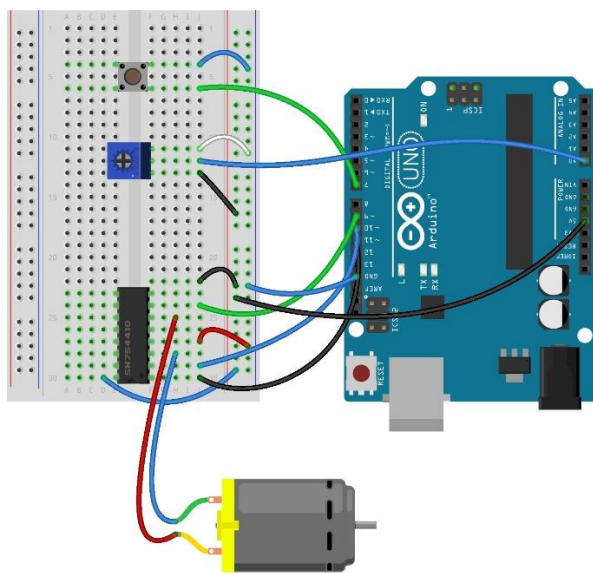
This was a build to control a DC motor so that a potentiometer controlled the speed of the motor and a switch controlled the direction the motor spun in. This build contains a trim pot, a push switch, an h-bridge, a dc motor and about 15 wires.

Issues:

I didn't really run into any issues with this build. There was a great tutorial that I followed so everything was simple to do. I was really impressed by how fast you could get the motor to move. I learned a bit more about how switches function and how the dc motor works but the most interesting thing I learned about was the h-bridge.

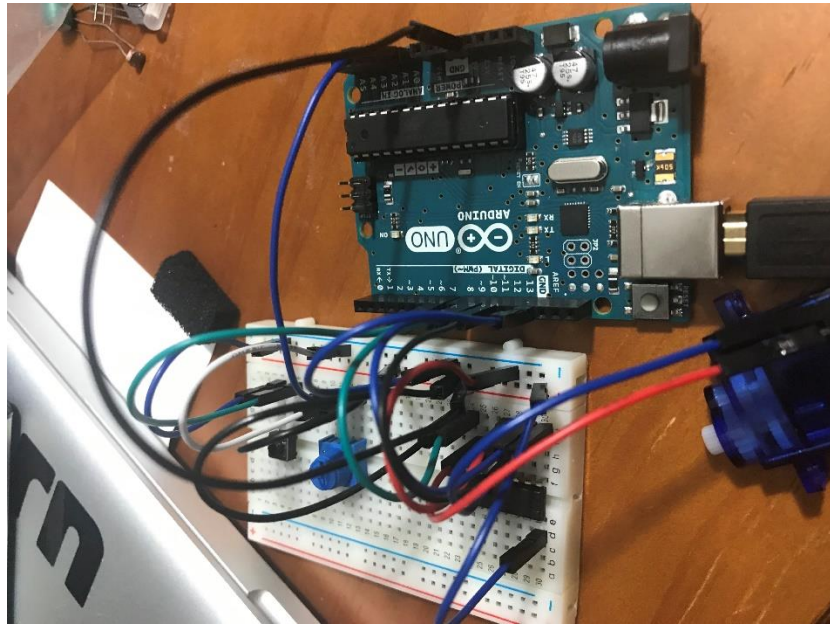
I was surprised by how the In1 and In2 pins kind of acted like a circuit where one pin needed to be connected to power in some way and the other one needed to be connected to ground. I kind of wished I learned more about why that was through this project.

Diagram of Build



fritzing print

Photo of Build



Code for Build

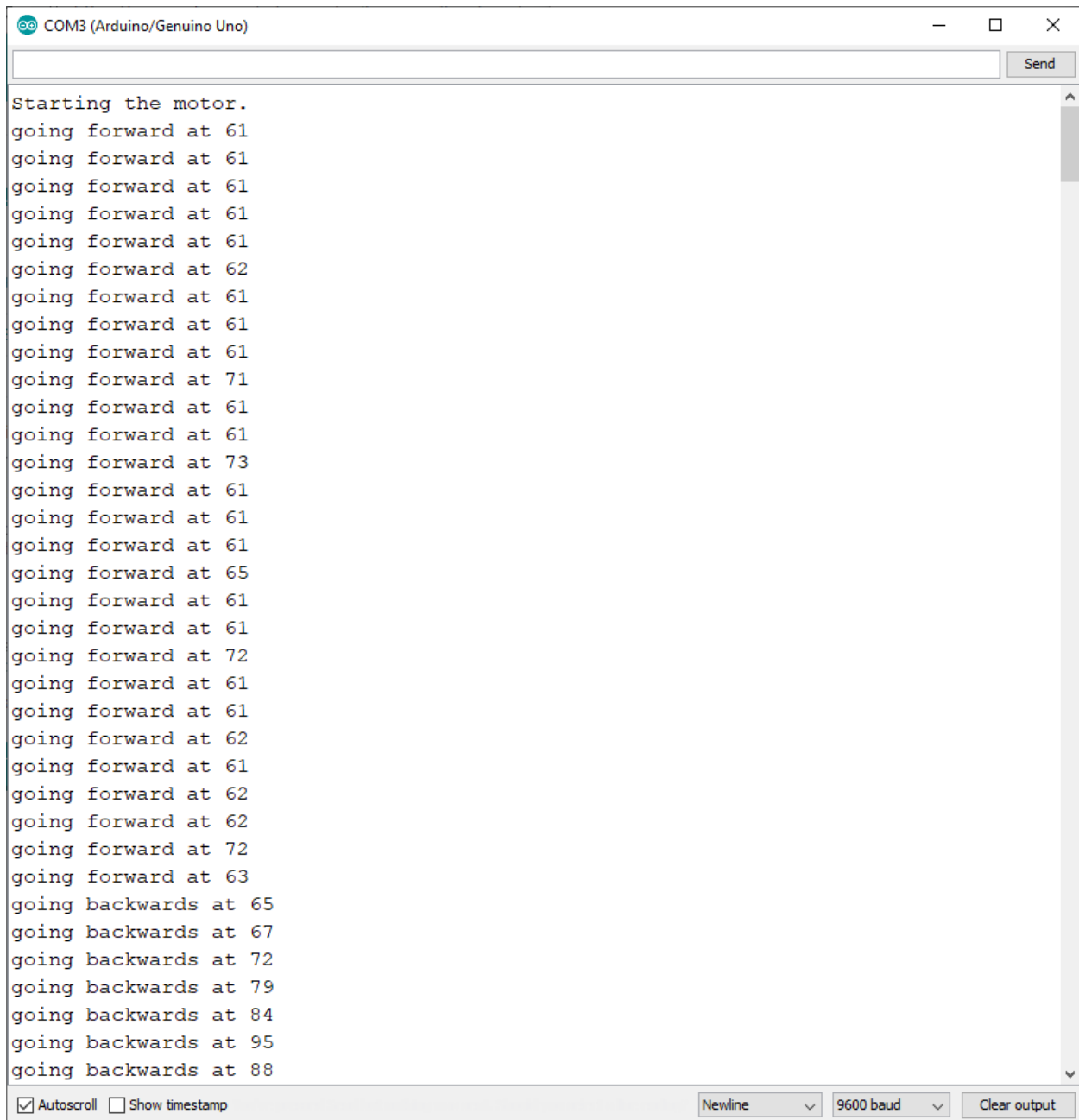
```
/*  
Code from https://learn.adafruit.com/adafruit-arduino-lesson-15-dc-motor-reversing/arduino-code.  
*/  
  
int enablePin = 11;  
int in1Pin = 10;  
int in2Pin = 9;  
int switchPin = 7;  
int potPin = A0;  
  
void setup()  
{  
  Serial.begin(9600);  
  Serial.println("Starting the motor.");  
  pinMode(in1Pin, OUTPUT);  
  pinMode(in2Pin, OUTPUT);  
}
```

```
pinMode(enablePin, OUTPUT);
pinMode(switchPin, INPUT_PULLUP);
}

void loop()
{
    //int speed = analogRead(potPin) / 4;
    int speed = analogRead(potPin);
    speed = map(speed, 0, 1023, 0, 255);
    boolean reverse = digitalRead(switchPin);
    setMotor(speed, reverse);
}

void setMotor(int speed, boolean reverse)
{
    if(reverse) {
        Serial.print("going forward at ");
        Serial.println(speed);
    } else {
        Serial.print("going backwards at ");
        Serial.println(speed);
    }
    analogWrite(enablePin, speed);
    digitalWrite(in1Pin, ! reverse);
    digitalWrite(in2Pin, reverse);
}
```

Example of Output



The screenshot shows the Arduino IDE serial monitor window for COM3 (Arduino/Genuino Uno). The window has a title bar with standard OS controls and a 'Send' button. The main area displays a list of commands and status messages. The output is as follows:

```
Starting the motor.  
going forward at 61  
going forward at 61  
going forward at 61  
going forward at 61  
going forward at 61  
going forward at 62  
going forward at 61  
going forward at 61  
going forward at 61  
going forward at 71  
going forward at 61  
going forward at 61  
going forward at 73  
going forward at 61  
going forward at 61  
going forward at 61  
going forward at 65  
going forward at 61  
going forward at 61  
going forward at 72  
going forward at 61  
going forward at 61  
going forward at 62  
going forward at 61  
going forward at 62  
going forward at 62  
going forward at 72  
going forward at 63  
going backwards at 65  
going backwards at 67  
going backwards at 72  
going backwards at 79  
going backwards at 84  
going backwards at 95  
going backwards at 88
```

At the bottom of the window, there are checkboxes for 'Autoscroll' (checked) and 'Show timestamp' (unchecked). To the right are dropdown menus for 'Newline' and '9600 baud', and a 'Clear output' button.

Build 2: Control a Stepper Motor

Description:

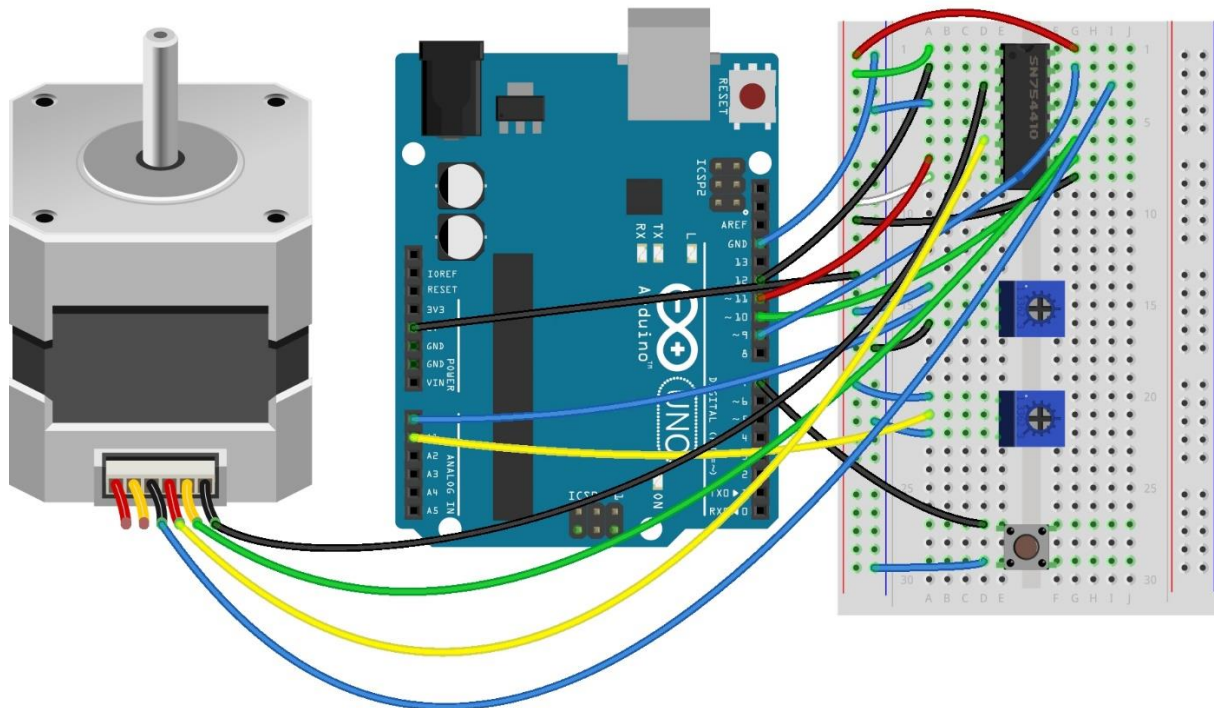
For this project, we had to control a stepper motor with two potentiometers and a switch. The idea was that the switch would control whether it would go forward or in reverse, one potentiometer controlled how fast the motor turned, and the last potentiometer controlled how many steps the motor turned. This build contained 2 trim pots, 1 switch, an h-bridge, and about 22 wires.

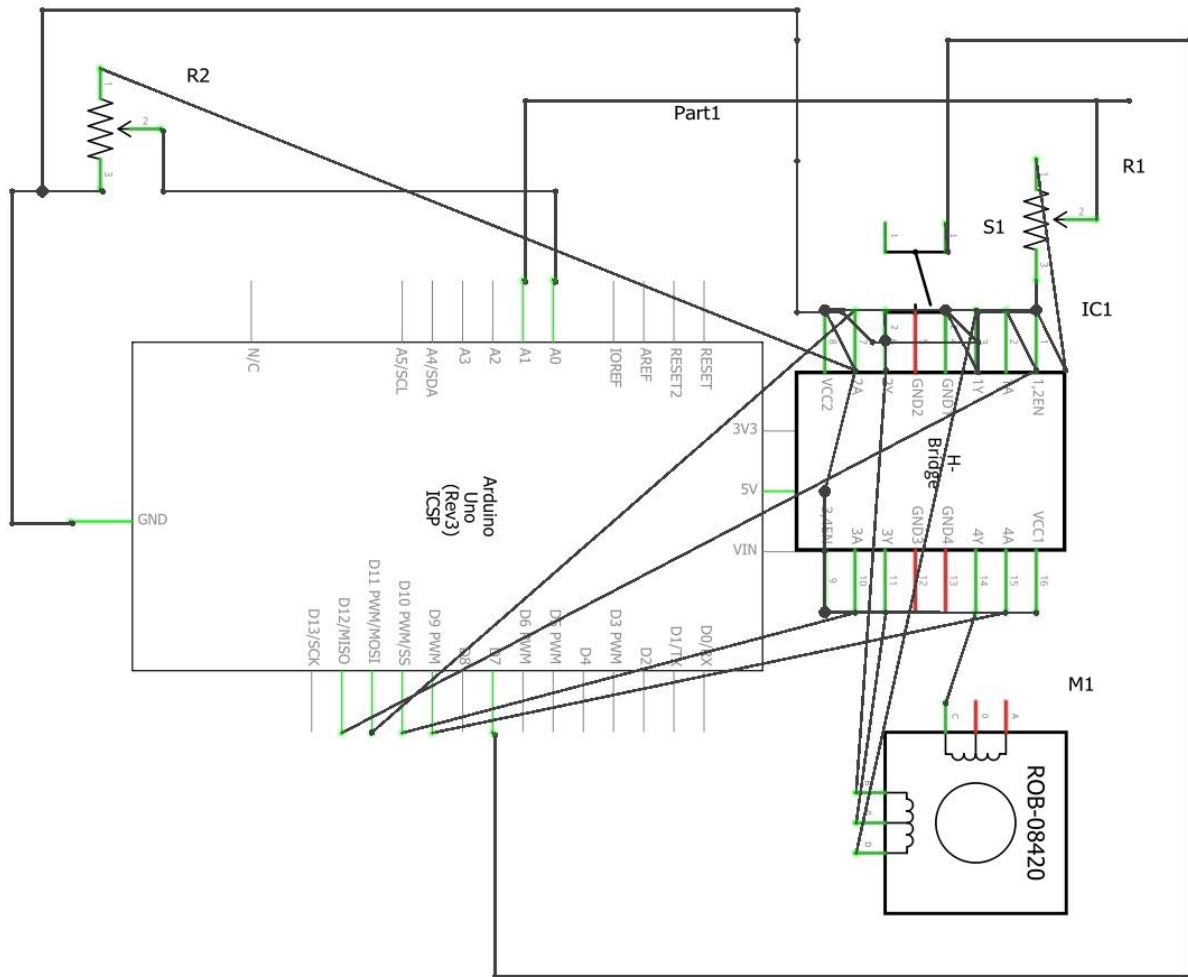
Issues:

So, the first issue I ran into with this build was trying to map the speed and steps of the potentiometers to the speed and step abilities of the motor. For a while, the motor was moving so slowly that nothing would happen, which made it difficult to figure out what was wrong. I took a step back and used the original code to figure out what the max speed was before it stopped moving and it turned out to be 50.

Once I mapped the potentiometer's output to a scale from 1 to 50, it started working in such a way that I could continue with the project. I was shocked at how small the speed range was and how I couldn't find any information on the limits when I was initially looking at the information from adafruit's website. After that, the build came together well. This build further cemented the idea that while available code can be helpful, it sometimes impairs my ability to do the project well because it doesn't quite work with my equipment. Overall, it made me feel a little more confident in my ability to work with this equipment.

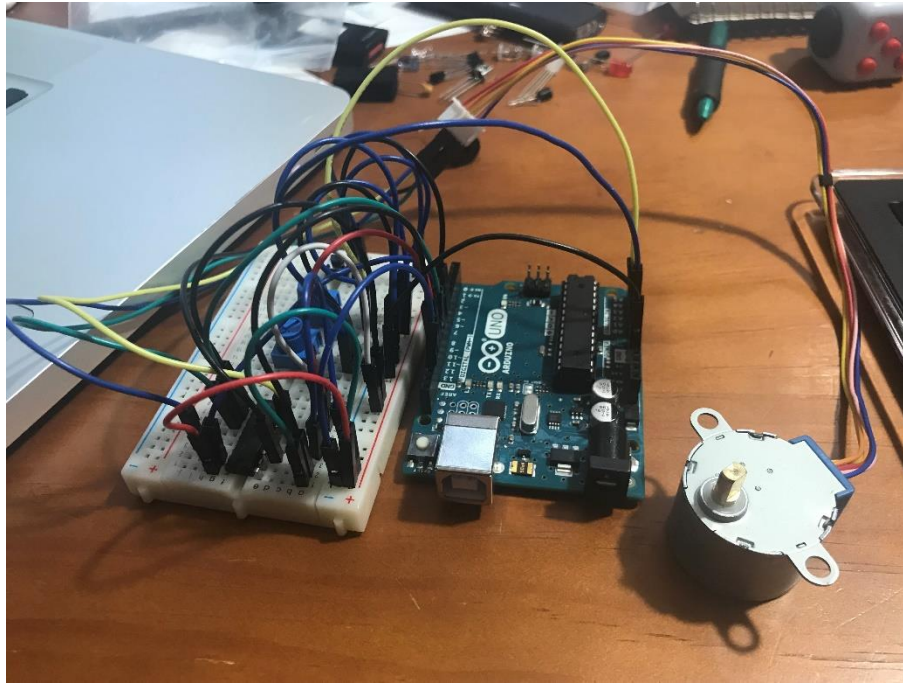
Diagram of Build





fritzing

Photo of Build



Code for Build

```
/*  
  
Code    from    https://learn.adafruit.com/adafruit-arduino-lesson-16-stepper-motors/arduino-code.  
  
Edited to deal with potentiometers and the switch that controls direction.  
  
Includes code from https://learn.adafruit.com/adafruit-arduino-lesson-15-dc-motor-reversing/arduino-code.  
  
*/  
  
#include <Stepper.h>  
  
const int stepsPerRevolution = 512; //number of steps per revolution for  
the motor  
  
int in1Pin = 12;
```

```
int in2Pin = 11;
int in3Pin = 10;
int in4Pin = 9;
int switchPin = 7;
int speedPin = A0;
int stepPin = A1;

int previousSpeed = 0;
int previousSteps = 0; // to how the motor's previous position
int speed = 0;
int steps = 0;

Stepper motor(stepsPerRevolution, in1Pin, in2Pin, in3Pin, in4Pin);

void setup()
{
    pinMode(in1Pin, OUTPUT);
    pinMode(in2Pin, OUTPUT);
    pinMode(in3Pin, OUTPUT);
    pinMode(in4Pin, OUTPUT);
    pinMode(switchPin, INPUT_PULLUP);
    pinMode(speedPin, INPUT);
    pinMode(stepPin, INPUT);

    Serial.begin(9600);
    Serial.println("Starting the motor.");
}

void loop()
{
```



```

    speed = analogRead(speedPin);
    speed = map(speed, 0, 1023, 1, 50);
    Serial.print("The speed is now: ");
    Serial.println(speed);
    motor.setSpeed(speed);
    steps = analogRead(stepPin);
    steps = map(steps, 0, 1023, 0, 512);
    boolean reverse = digitalRead(switchPin);
    driveMotor(steps, reverse);
    previousSpeed = speed;
    delay(5000);
}

void driveMotor(int steps, boolean reverse)
{
    if(steps != abs(previousSteps)) {
        Serial.println("Resetting to previous position.");
        motor.setSpeed(previousSpeed);
        motor.step(-1 * previousSteps);
        delay(1000);
        motor.setSpeed(speed);
    }

    if(reverse) {
        Serial.println("going in reverse.");
        Serial.print("going at ");
        Serial.print(steps);
        Serial.println(" steps.");
        steps = -1 * steps;
        motor.step(steps);
    }
}

```

```
    } else {  
        Serial.println("going forwards.");  
        Serial.print("going at ");  
        Serial.print(steps);  
        Serial.println(" steps.");  
        motor.step(steps);  
    }  
  
    previousSteps = steps;  
}
```

Example of Output

COM3 (Arduino/Genuino Uno)

Send

Starting the motor.
The speed is now: 49
Resetting to previous position.
going in reverse.
going at 158 steps.
The speed is now: 49
Resetting to previous position.
going in reverse.
going at 159 steps.
The speed is now: 48
Resetting to previous position.
going in reverse.
going at 266 steps.
The speed is now: 48
going forwards.
going at 266 steps.
The speed is now: 48
going forwards.
going at 266 steps.
The speed is now: 49
Resetting to previous position.
going in reverse.
going at 267 steps.
The speed is now: 49
Resetting to previous position.
going in reverse.
going at 384 steps.
The speed is now: 49
going in reverse.
going at 384 steps.
The speed is now: 49
Resetting to previous position.
going in reverse.
going at 385 steps.
The speed is now: 40
Resetting to previous position.
going in reverse.
going at 384 steps.
The speed is now: 34
Resetting to previous position.
going in reverse.
going at 382 steps.
The speed is now: 34
going forwards.
going at 382 steps

☒ Autoscroll ☐ Show timestamp

Newline 9600 baud Clear output